$2.00

DIGITAL EQUIPMENT CORPORATION

education

Populution: A Self-Teaching BASIC Primer

computers are for kids — digital — EduSystems—expandable, economical

POPULUTION: A PRIMER

A SELF-TEACHING BASIC WORKBOOK

Copyright © 1972 by Digital Equipment Corporation

For additional copies, order DEC-E8-ZJZA-D from DEC, Direct
Mail, Building 1-1, Maynard, Mass. 01754. Price $2.00

CONTENTS

CHAPTER 1. SAY SOMETHING IN BASIC................ 1

GETTING STARTED 1
SMALL TALK 3
MISTAKES 8

CHAPTER 2. GROWING PAINS........................ 9

SHORTHAND FOR LARGER NUMBERS 9
TOO MANY PEOPLE 11
BOXES 13
THE PRINT STATEMENT 15

CHAPTER 3. AUTOMATION........................... 17

DIVISION OF LABOR 17
SLOW DOWN, TEACHER! 20
READ AND DATA 23

CHAPTER 4. COMPUTER POWER!...................... 25

TABLE-MAKER 25
CROWDED 29
BEWARE OF MATHEMATICAL MODELS 32

CHAPTER 5. WORLDS OF IF......................... 33

SON OF TABLE-MAKER 33
RACE TO OBLIVION 37
YOUR TURN 40

CHAPTER 6. CAROUSEL............................. 41

1, 2, 3 41
FOR AND NEXT EXPLAINED 45
YOUR TURN 48

CHAPTER 7. A PLACE FOR EVERYTHING............... 49

SUBSCRIPTED VARIABLES 49
GOSUB 53

CHAPTER 8. DEMOGRAPHY........................... 57

DEMOGRAPHY? 57
CHANGING THE DATA BASE 61
YOUR TURN...LAST TIME! 63
ON BECOMING FLUENT 64

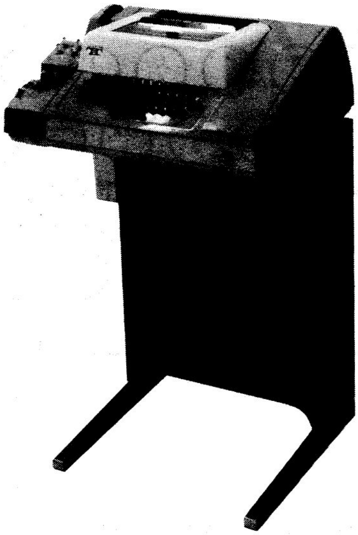
CHAPTER 1

SAY SOMETHING IN BASIC

GETTING STARTED

A computer has no eyes, no ears ... a computer has no mouth, no vocal cords. So how do we communicate with a computer?

We will use a *teletypewriter*.



Teletypewriter

The teletypewriter is connected to the computer.

→ Hardwired, by means of a cable that directly connects the teletypewriter to the computer. Look for the cable ... follow it ... find the computer! (But don't confuse the cable with the power cord.)

OR

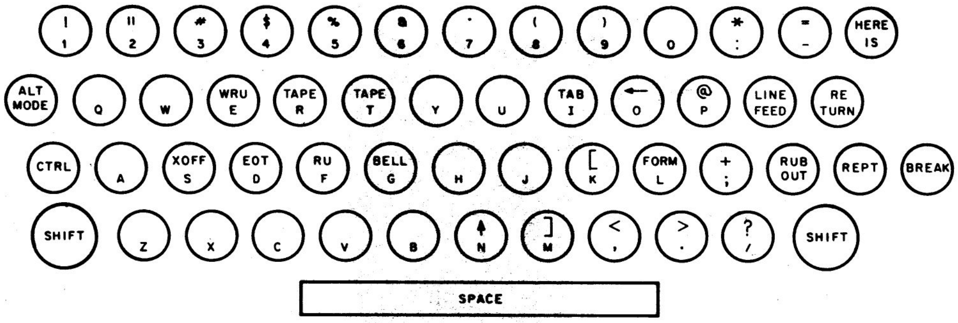
→ Indirectly, using the telephone system as the link. In this case, the teletypewriter may be remote from the computer ... it can be in any place that has telephone service.

> REMEMBER THIS. Many people use "TTY" as an abbreviation for "teletypewriter." We will too.

We use the TTY (teletypewriter ... remember?) to send messages to the computer. In turn, the computer sends messages to the TTY for us to read.

We send our messages in a language called BASIC. That's what this book is all about - to help you learn to converse with a computer, using a language called BASIC.

Let's begin. Here is a diagram of the TTY keyboard.



Locate the keys with the following symbols.

Letters:	A	B	C	D	E	F	G	H	I	J	K	L	M
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Digits:	1	2	3	4	5	6	7	8	9	0			
Special:	+	-	*	/	↑	=	.	,	;	(	)	←	"

To type a letter or a digit, simply press the appropriate key.

To type . or , or : or / or -, press the appropriate key.

To type one of these hold down either SHIFT key and press the key with the desired character.

+ * ↑ ( ) ← " =

SMALL TALK

Begin! Type something on the TTY and press the RETURN key. The computer will probably type

```
WHAT?
```

The message varies from computer to computer. But the point is ... the computer did not understand you.

or perhaps

```
???
```

or maybe

```
ERROR IN LINE 0
```

So try this ... *you* press the keys ... *you* do the typing.

> Finish each line by pressing
>
> RETURN

Do it. You type

```
SCR

10  PRINT "MY HUMAN UNDERSTANDS ME"
20  END
```

Don't forget to press the RETURN key.

Then type

```
RUN
```

and the computer will type

```
MY HUMAN UNDERSTANDS ME
```

Nice thought, but it wasn't an original computer idea. The computer simply printed what you told it to print.

Let's review what happened. First, you typed

```
SCR
```

and pressed the RETURN key. This told the computer to SCRatch any old data in its memory. It's sort of like erasing the blackboard before you start writing on it.

Next, you typed a short *program*, consisting of two *statements*.

```
10  PRINT "MY HUMAN UNDERSTANDS ME"
20  END
```

Each statement begins with a line number (10, 20).

As you typed the program, it was stored into the computer's memory. After entering the program, you then typed

```
RUN
```

The computer ran the program and typed

```
MY HUMAN UNDERSTANDS ME
```

then it stopped.

The statement: PRINT "MY HUMAN UNDERSTANDS ME"

tells the computer to print this message.

The statement: END

tells the computer to stop.

Every statement must begin with a *line number*. A line number is a positive integer between 1 and 9999, inclusive.

To run a program, type RUN.

To erase a program, type SCR.

Still your turn. Try this one ... type

```
SCR                          SCRatch the preceding program

10  PRINT "7 + 5"            Enter the new program
20  END

RUN                          RUN the new program

7 + 5 <--------------------- The computer types what you tell
                             it to type.
```

Next ... let's replace line 10 with a new line 10.

```
10  PRINT 7 + 5              No quotation marks.
```

Then tell the computer to LIST the current program.

```
LIST
10  PRINT 7 + 5 <----------- Here is the new line 10
20  END <------------------> and the old line 20

RUN                          RUN it.

 12                          This time the computer does the
                             arithmetic.
```

The statement

```
PRINT 7 + 5
```

tells the computer to evaluate the numerical expression 7 + 5 (that is, do the arithmetic) and print the result as a decimal numeral.

The statement

```
PRINT "7 + 5"
```

tells the computer to print the string enclosed in quotation marks *exactly as it appears*. No arithmetic is performed.

Strings? Numerical expressions?

```
PRINT  "MY HUMAN UNDERSTANDS ME"
```

This is a string. It is enclosed in quotation marks.

```
PRINT "7 + 5"
```

This is a string.
It is enclosed in
quotation marks.

```
PRINT 7 + 5
```

This is not a
string. It is
a numerical ex-
pression.

Your turn again. Try these.

```
SCR

10  PRINT "7 + 5 =" , 7 + 5
20  END
RUN

7 + 5 =         12
```

Note the comma between "7 + 5 =" and 7 + 5.

Replace line 10 as follows.

```
10  PRINT "7 + 5 =" ; 7 + 5
```

Note the semi-colon between "7 + 5=" and 7 + 5.

Then LIST the modified program and RUN it.

```
LIST
10  PRINT "7 + 5=" ; 7 + 5
20  END
RUN

7 + 5 = 12
```

If a PRINT statement contains more than one item (string or expression) the items must be separated by commas or semi-colons.

Tell the computer to

```
SCR
10  PRINT 7+5, 7-5, 7*5, 7/5
20  END
RUN

 12          2             35            1.4
```

```
To tell the computer to add, use      +
To tell the computer to subtract, use -
To tell the computer to multiply, use *
To tell the computer to divide, use   /
```

To tell the computer to squeeze the answers more closely together, use semicolons instead of commas.

```
10  PRINT 7+5 ; 7-5 ; 7*5 ; 7/5
RUN

 12   2   35   1.4
```

Mixed operations? Try this one.

```
10  PRINT 2*3+4, 2*3 + 4*5, 2*3/4
RUN

 10          26            1.5
```

Use parentheses to group terms.

```
10  PRINT 2*(3 + 4), (2 + 3)*(4 + 5), (2 + 3)/(4 + 5)
RUN

 14          45             .555556
```

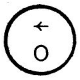
MISTAKES

Do you occasionally make mistakes? We do ... watch.

10 PTINT 2*3+4	We misspell PRINT.
SYNTAX ERROR	The computer tells us we made an error. (Look up "SYNTAX" in your dictionary.)

The error message may be different on your computer. That's not the point. The point is: if we had noticed that we hit T when we meant to hit R, we could have corrected it by using the back arrow.

The back arrow ← is above the letter O.

To type ← hold SHIFT key down and press 

PT←RINT 2*3+4	The back arrow (←) deletes the character that it points to.
PTTT←←←RINT 2*3+4	Three wrong characters wiped out by three back arrows.

> Can I do this? Can I do that? What happens if I
>
> ____?
>
> (You complete the question)
>
> Obviously, we can't answer all your questions in this book. But *you* and the computer can answer most of them.
>
> EXPERIMENT! ... GAMBLE! ... GUESS, THEN TRY IT!

CHAPTER 2

GROWING PAINS

SHORTHAND FOR LARGE NUMBERS

The population of the U. S. is about 205 MILLION people.

205 MILLION = 205,000,000

Tell the computer to print 205,000,000.

```
10  PRINT 205000000          No commas!  Why?  Try it and find out.
99  END
RUN

 2.050000E+8                 This is a floating point number.
                             Read on!
```

Floating point notation is simply a shorthand way of expressing very large or very small numbers. In floating point notation a number is represented by a *mantissa* and an *exponent*. They are separated by the letter E.

$$\underbrace{2.050000}_{\text{mantissa}}\text{E}\underbrace{+8}_{\text{exponent}}$$

The exponent tells you how to move the decimal point of the mantissa in order to obtain the ordinary notation form of the number. If the exponent is positive, move the decimal point of the mantissa to the right. If the exponent is negative, move the decimal point to the left. (What if the exponent is zero?)

Here are some more examples showing numbers expressed in floating point notation and "ordinary notation."

FLOATING POINT NOTATION	ORDINARY NOTATION	
1.000000E+6	1000000	(1 MILLION)
1.000000E+9	1000000000	(1 BILLION)
1.000000E+12	1000000000000	(1 TRILLION)
1.000000E-6	.000001	(1 MILLIONTH)
1.000000E-9	.000000001	(1 BILLIONTH)
1.000000E-12	.000000000001	(1 TRILLIONTH)

Perhaps you have noticed that the computer always prints the mantissa as a seven digit number and the exponent as a one or two digit number.

An experiment.

```
10  PRINT 123456 , 1234567 , 12345678
99  END
RUN

 123456          1.234567E+6      1.234568E+7
```

The mantissa is rounded to seven digits.

Your turn.

A light year is a measure of distance, the distance that light travels in one year. The speed of light is 186283 miles per second. How many miles in a light year? Use the computer to find out and write the answer below in

floating point notation ____________________ miles.

ordinary notation ________________________ miles.

If a snail slithers at the rate of one inch per second, how long will it take him to slither completely around the earth, following the equator? (Assume that the snail can slither on water.) The *radius* of the earth is about 4000 miles and $C = 2\pi r$.

Your answers __________ = __________ = __________
seconds days years

TOO MANY PEOPLE

At the end of 1970, the population of the earth was about 3.6 BILLION people.

3.6 BILLION = 3,600,000,000 = 3.6E9

If the present growth rate persists, the population will double every 35 years. Suppose this actually happens ... what will the population be in the year 2250?

$$\frac{2250 - 1970}{35} = \frac{280}{35} = 8 \text{ doublings}$$

We could do it like this.

```
    10  PRINT 3.6E9*2*2*2*2*2*2*2*2        (8 doublings ... count them!)
    99  END
    RUN

     9.216000E+11 <-------------------------- Too many people!
```

Here is a shorter way.

Do you remember? $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8$

In BASIC, we write 2^8 like this: 2↑8

```
    10  PRINT 3.6E9*2↑8                     Multiply 3.6E9 by 2^8.
    99  END
    RUN

     9.216000E+11 <-------------------------- Still too many people!
```

Remember

- To compute a *power* of a number, use ↑

All those people ... where will they live? On the surface of the earth. But what is the surface area of the earth? We don't know - but we do know this:

→ The earth is very nearly spherical in shape.
→ Its radius is about 4000 miles.
→ The surface area of a sphere is

$$S = 4\pi r^2$$

"approximately equal to"

where r is the radius and $\pi \overset{\swarrow}{\cong} 3.14$

Therefore, the surface area of the earth is

$$S \cong 4 \times 3.14 \times 4000^2 \text{ square miles.}$$

Use the computer.

```
10  PRINT 4*3.14*4000↑2
99  END
RUN
 2.009600E+8
```

The surface area of the earth is

2.009600E+8 = 200960000 square miles.

You may prefer a more verbal approach.

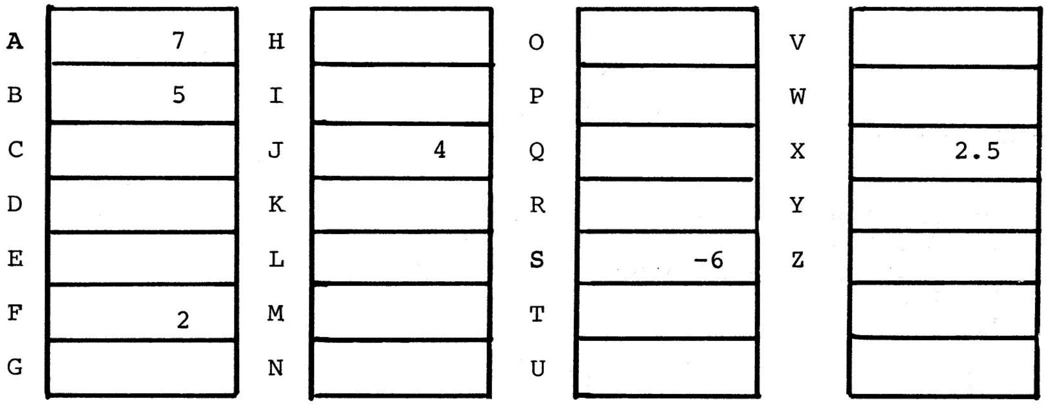
```
10  PRINT "AREA OF EARTH IS" ; 4*3.14*4000↑2 ; "SQUARE MILES"
99  END
RUN

AREA OF EARTH IS 2.009600E+8 SQUARE MILES
```

How many people per square mile in

1970?____________2205__________

BOXES

Deep down inside the computer there are 26 little boxes.



A	7	H		O		V	
B	5	I		P		W	
C		J	4	Q		X	2.5
D		K		R		Y	
E		L		S	-6	Z	
F	2	M		T			
G		N		U			

26 BOXES LABELED A THROUGH Z

Each box can contain one number at any one time. We have already stored numbers in some of the boxes.

- 7 is in box A
- 5 is in box B

What number is in box F? __________ in J? ___________

-6 is in box _________ and 2.5 is in box ___________.

O.K., using a *pencil*, put 8 into C. In other words, write the numeral '8' in the box labeled 'C.' Then do the following, carefully!

(1) Put 12 into N.

(2) Put 27 into N. But wait! A box can hold only one number at a time - before you can enter 27 into N, you must first *erase* the 12 that you had previously entered.

When the computer puts a number into a box, it *automatically* erases the previous content of the box.

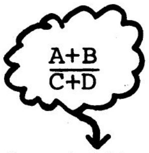
Tell it to the computer.

```
10  LET A = 7<————————————————————Put 7 into box A.
20  PRINT A<——————————————————————Print the content of box A
99  END
RUN

 7
```

Another example.

```
10  LET A = 7
20  LET B = 5
30  PRINT A+B, A-B, A*B, A/B
99  END
RUN

 12            2          35        1.4
```

More practice? O.K.



```
10  LET A = 2
20  LET B = 3
30  LET C = 4
40  LET D = 5
50  PRINT A+B+C+D, A*B*C*D, A*(B+C), (A+B)/(C+D
99  END
RUN

 14            120          14       .5555556
```

We call A, B, C, ..., Z *variables* and refer to the number in box A as the *value* of A. The number in box B is the value of B, the number in C is the value of C and so on.

Without using the computer, complete each of the following RUNs as you think the computer would do it. Then use the computer to find out if you are correct.

```
10  LET A=1
20  LET A=2
30  PRINT A
99  END
RUN

__________
```

```
10  LET A=7
20  LET B=A
30  PRINT B
99  END
RUN

__________
```

```
10  LET A=1
20  PRINT A
30  LET A=2
40  PRINT A
99  END
RUN
__________

__________
```

THE PRINT STATEMENT

We have used several PRINT statements. The general form of a PRINT statement is shown below.

PRINT *list*

where *list* is a list of *expressions* and/or *strings*. Items in the list must be separated by commas or semicolons.

Expressions come in two flavors, numerical and variable.

Numerical expression: 2*(3+4)
Variable expression: A*(B+C)

More examples? O.K.

NUMERICAL EXPRESSIONS	VARIABLE EXPRESSIONS
2*3 + 4*5	A*B + C*D
4*3.14*400↑2	4*3.14*R↑2
3.6E9/2.0096E8	P/S

A string is a set of symbols enclosed in quotation marks.

PRINT "A*B"

This is a string.
It is enclosed in
quotation marks.

PRINT A*B

This is *not* a
string. It is a
variable expression.

The statement PRINT A*B tells the computer to evaluate the variable expression A*B using the *current values* of A and B (the numbers in boxes A and B) and print the answer.

Store the following program and use the computer to help you complete the table. Modify the program as you wish.

(Hint: Change Line 10.)

N is the number of doublings.

```
10  LET Y=2005
20  LET N=(Y-1970)/35
30  LET P=3.600000E+09*2↑N
40  LET A=2.009600E+08
50  PRINT " YEAR"," PEOPLE","PEOPLE/SQ. MI."
60  PRINT Y,P,P/A
99  END
RUN

 YEAR          PEOPLE        PEOPLE/SQ. MI.
 2005          7.200000E+09   35.82803
```

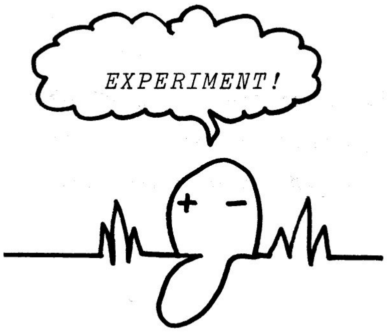
YEAR	PEOPLE	PEOPLE/SQ. MI.
1970	3.600000E+9	17.91401
2005	7.200000E+9	35.82803
2040		
2075		
2110		
2145		
2180		

If the population continues to double every 35 years, what will the population be in the year 3020?

Your answer: ______________________________

How many people per *square yard* in 3020?! (There are 1760 yards in one mile.)

Your answer: ________________________


EXPERIMENT!

CHAPTER 3

AUTOMATION

DIVISION OF LABOR

You and the computer have worked diligently. But you have done too much of the work, the computer too little.

A PROBLEM. In year zero, we start with a population of P people. The population increases by 1% each year. In n years the population will be

$$\underset{\uparrow\ \text{population at the end of n years}}{Q} = \underset{\uparrow\ \text{initial population}}{P}(1 + \overset{\text{1\% increase per year}\ \downarrow}{1/100})^{n \longleftarrow \text{n years}}$$

If the growth rate is 2% per year, then

$$Q = P(1 + 2/100)^n$$

If the growth rate is 2.5% per year, then

$$Q = P(1 + 2.5/100)^n$$

and if the growth rate is r% per year, then

$$Q = P(1 + r/100)^n$$

Let's write a BASIC program to compute and print the value of Q for given values of P, R, and N. WHEN YOU LOOK AT THE PROGRAM, REMEMBER THESE THINGS.

P is the initial population.
R is the growth rate.
N is the number of years.
Q is the population in N years.

You do it. First, type

```
SCR
```

and press the RETURN key.

```
10  INPUT P
20  INPUT R
30  INPUT N
40  LET Q = P*(1 + R/100)↑N
50  PRINT Q
60  END
```

Then enter the program. Enter each statement exactly as shown (and press the RETURN key after each statement of course).

The program is now stored in the computer's memory. Now you can RUN it. Type

```
RUN
?
```

and press RETURN. The computer types a question mark.

The computer wants something. It wants what the program tells it to want. It wants you to INPUT P. Do it. Enter 1000 as the initial population.

```
? 1000

?
```

Enter 1000 and press RETURN. The computer types another question mark.

Now what does it want? Of course, the value of R. Do it!

```
? 1000
? 1
?
```

Enter 1 (for 1%) and press RETURN. The computer types another ?.

That's right. The computer wants the value of N. Let's try 20.

```
?1000
? 1
? 20

 1220.190
```

Enter 20 and press RETURN. The computer computes and prints the value of Q, then stops as it reaches and obeys the END statement.

The program is still in the computer's memory. RUN it again.

```
RUN
? 1000
? 2
? 35

 1999.889
```

P = 1000 people - initial population
R = 2% increase each year.
N = 35 years.

Q is about 2000 people. The population doubled!

To RUN the program

- → Type RUN and press the RETURN key.
- → The computer types a question mark. Type the value of P and press the RETURN key.
- → The computer types a question mark. Type the value of R and press the RETURN key.
- → The computer types a question mark. Type the value of N and press the RETURN key.

Do it!

```
RUN
? 3.6E9                  P = 3.6 billion people (Earth, 1970)
? 2                      R = 2%
? 100                    N = 100 years

 2.608064E+10            Q = too many people!
```

Your turn. Do one for us. Complete the following.

```
RUN
? 200E6                  P = 200 million people (USA, 1970)
? 1.5                    R = 1.5%
? 100                    N = 100 years

____________________     Q = ____________________ people
```

Then do one for yourself. You pick the values of P, R and N.

```
RUN
? ___                    Your value of P.

? ___                    Your value of R.

? ___                    Your value of N.

_______________          Q = ____________________ people.
```

SLOW DOWN, TEACHER!

O.K. Let's retrace our steps. We start with a problem.

THE PROBLEM: Compute Q for given values of P, r and n.

$$Q = P(1 + r/100)^n$$

We write a program. Then, program in hand, we cautiously approach the computer, confront the TTY, type

```
SCR                           and press RETURN.
```

The computer clears its memory of any old programs and patiently awaits our next move. We grow bold ... throwing caution to the wind, we type in the program. Observe!

```
10  INPUT P
20  INPUT R
30  INPUT N
40  LET Q = P*(1 + R/100)↑N
50  PRINT
END
```

Then, triumphantly, we type

```
RUN                           and press RETURN.
```

The rest is history. The computer RUNS the program, periodically requesting data. It types a question mark; we type the value of P. It types a question mark; we type the value of R. It types a question mark; we type the value of N. Then, having obtained the given data, the computer quickly, efficiently, effortlessly, flawlessly, computes the value of Q, prints it and stops.

Now that we have solved the problem one way, let's do it again using a slightly different approach.

As usual, we want you to do it. Type, as directed below.

```
SCR                                First, SCRatch the old program.

10  INPUT P, R, N               ┐
20  LET Q = P*(1 + R/100)↑N     ├── Enter the new program.
30  PRINT Q                     │
99  END                         ┘
RUN                                Then RUN the new program.

? 1000, 1, 20                      Enter numbers for P, Q, R and press
 1220.190 <──────────────────────  RETURN.  Here is the answer.
```

In the above program there is only *one* INPUT statement, but it asks for input of *three* numbers, one number for each variable in the INPUT statement.

```
10  INPUT P, R, N
           ↗  ↗  ↗
?  1000, 1, 20
```

Let's look at the general form of an INPUT statement.

GENERAL FORM:	*line number*	INPUT	*list of variables*
EXAMPLE:	10	INPUT	P, R, N

The variables in the list are *separated* by commas. That is, between any two variables, there is a comma. But note carefully there is <u>no</u> comma following INPUT and there is <u>no</u> comma following the last variable in the list.

- → INPUT tells the computer to type a question mark and then stop. It's your move.
- → You must enter a number for each variable in the list of variables in the INPUT statement. Type commas between numbers.
- → After typing the last number, press RETURN.

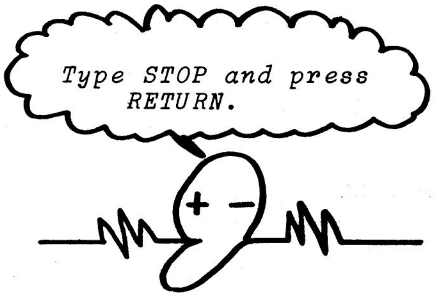
Is the program on Page 21 still in the computer? If so, good. If not, enter it.

Now we assume that the program on Page 21 *is* in the computer. Do not type SCR. Instead, type

```
40  GO TO 10                New statement.  It tells the
                            computer to GO TO Line 10.
```

You have *added* a statement to the program in the computer. Let's look at the program as it now resides in the computer's memory. Type

```
LIST                                and press RETURN.  The computer
                                    now types the complete program,
                                    like this.


10  INPUT P, R, N  <-----------------------
20  LET Q = P*(1 + R/100)↑N
30  PRINT Q
40  GO TO 10 <----------------------Here is the statement you added,
99  END                             right where it should be, between
                                    Line 30 and Line 99.
RUN

? 1000, 1, 20                       Enter values.
 1220.19                            Computer prints answer and goes around.
? 1000, 2, 35                       Enter values.
 1999.889                           Computer prints answer and goes around.
? 3.6E9, 2, 100                     Enter values.
 2.608072E+10                       Computer prints answer and goes around.

?____________                       Your turn.  Enter values

_____________                       Computer prints answer and goes around.

?                                   But how do we get out of this?
```


Type STOP and press RETURN.
+ −

```
? STOP

READY                               Computer types READY.  Read on!
```

READ and DATA

Let's start with the program on the preceding page and make a couple of changes.

```
10  READ P, R, N                        Change INPUT to READ
30  PRINT P, R, N, Q                    Change the PRINT statement.

90  DATA 1000, 1, 20                    Add three DATA statements.
91  DATA 1000, 2, 35                    Each DATA statement contains one
92  DATA 3.6E9, 2, 100                  set of data ... values of P, R, N.

LIST                                    Then LIST the modified program.

10  READ P, R, N <--------------------- Here is a change.
20  LET Q = P*(1 + R/100)↑N
30  PRINT P, R, N, Q <----------------- Here is a change
40  GO TO 10
90  DATA 1000, 1, 20    --+
91  DATA 1000, 2, 35      |------------ And three added statements
92  DATA 3.6E9, 2, 100  --+
99  END

RUN

 1000             1          20            1220.190
 1000             2          35            1999.889
 3.600000E+9      2          100           2.608072

OUT OF DATA IN LINE 10 <--------------- This means what it says.  The
                                        computer has run out of data.
```

Now it's your turn. Add the following statement,

```
5  PRINT " P", " R", " N", " Q"
```

and RUN the program. The results should look like the following.

```
P                R           N             Q
1000             1           20            1220.190
1000             2           35            1999.889
3.600000E+9      2           100           2.608072

OUT OF DATA IN LINE 10
```

If you want to see what the program looks like, LIST it.

We used three DATA statements, each with one set of values for P, R and N. But we could have put all the data in one DATA statement.

Try it. Type

```
90  DATA 1000,1,20,1000,2,35,3.6E9,2,100
```

Wait! Since you have put all the data in Line 90, you no longer need lines 91 and 92. Let's delete them.

```
91
92
```

If you type a line number and press RETURN, the computer erases that line, *but only that line*, from its memory.

Now LIST the program and then RUN it.

```
LIST
5  PRINT " P", " R", " N", " Q"
10  READ P, R, N
20  LET Q = P*(1 + R/100)↑N
30  PRINT P, R, N, Q
40  GO TO 10
90  DATA 1000,1,20,1000,2,35,3.6E9,2,100
99  END
RUN

 P               R               N               Q
 1000            1               20              1220.190
 1000            2               35              1999.889
 3.600000E+9     2               100             2.608072E+10

OUT OF DATA IN LINE 10
```

READ tells the computer to read numbers from a DATA statement, one number for each variable in the READ statement. Use as many DATA statements as you wish to hold the data. When the computer uses all the data in a DATA statement, it automatically moves to the next DATA statement. If it can't find any more data, it types a message such as OUT OF DATA.

> Try this. Add the statement   7 PRINT   and RUN the program again.

CHAPTER 4

COMPUTER POWER!

TABLE-MAKER

With the help of our trusty computer, we have constructed a table showing the value of

$$Q = P(1 + r/100)^n$$

for fixed values of P and r and several equally spaced values of n. Here is a copy of the computer RUN.

```
    RUN

    P=? 10000<———————————————————Initial population = 100000
    R=? 2  <—————————————————————Growth rate = 2% per year

                   POPULATION

     0              100000
     10             121899.4
     20             148594.7          Computer prints a table showing
     30             181136.1          population at the end of 0, 10,
     40             220803.9          20, 30, ... , 100 years.
     50             269158.8
     60             328103
     70             399955.7
     80             487543.8
     90             594313.1
     100            724464.4
```

and so on ... we stopped the computer by pressing Ⓢ

If we had not interrupted the computer, it would have continued on and on and on and....

Let's look at the program.

> Beware! If you RUN the program, the computer will not stop you following n = 100, *unless you stop it.* How?
>
> Press the Ⓢ key to stop the computer.

Here is the program that produced the RUN on the preceding page. We RUN it again, this time for P = 205E6 and r = 1%.

```
10  PRINT "P=";
15  INPUT P
20  PRINT "R=";
25  INPUT R
30  PRINT
40  PRINT " N","POPULATION"
45  PRINT
50  LET N=0
60  LET Q=P*(1+R/100)↑N
70  PRINT N,Q
80  LET N=N+10
90  GO TO 60
99  END
RUN

P=? 205E6
R=? 1

 N              POPULATION

 0               2.050000E+08
 10              2.264475E+08
 20              2.501389E+08
 30              2.763090E+08
 40              3.052170E+08
 50              3.371494E+08
 60              3.724226E+08
 70              4.113862E+08
 80              4.544263E+08
 90              5.019693E+08
 100             5.544863E+08
  .                   .
  .                   .
  .                   .
```

205E6 = 2.05E8

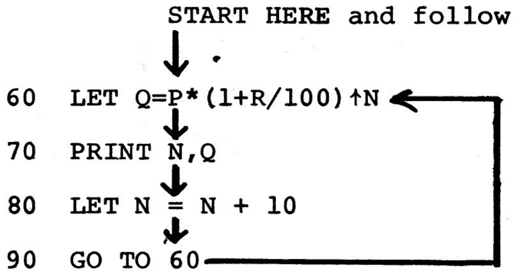
and so on ... we stopped the computer by pressing Ⓢ

RUN the program again for your values of P and r.

O.K., we'll try to explain how the program works.

THE STATEMENT(S) BELOW	CAUSE THE FOLLOWING PRINTOUT
10 PRINT"P="; 15 INPUT P	P=? (then you supply the value of P)
20 PRINT"R="; 25 INPUT R	R=? (then you supply the value of R)
30 PRINT	One vertical line space.
40 PRINT" N", "POPULATION"	N POPULATION
45 PRINT	One line space.
50 LET N = 0	No printout. The computer puts 0 into box N.
60 LET Q=P*(1+R/100)↑N 70 PRINT N,Q 80 LET N=N+10 90 GO TO 60	This is a *loop*. Lines 60 through 90 are repeated over and over. Each time through the loop, the computer prints one line of the table.
99 END	

Let's check the *loop* once more.



START HERE and follow the arrows

```
60   LET Q=P*(1+R/100)↑N
70   PRINT N,Q
80   LET N = N + 10
90   GO TO 60
```

Are you confused by LET N = N + 10? This statement tells the computer,

→ increase the value of N by 10, or

→ add 10 to the number in box N and
put the result into box N.

The new value of N *replaces* the old value of N.

One more look at the program. We will *trace* it. That is, we will follow the computer as it RUNs the program and record the values of P, R, N and Q after each statement is obeyed by the computer. Since the computer goes through the loop repeatedly, we will too. We will show what happens the first three times through the loop. (We assume P = 100000, R = 2.)

STATEMENT	P	R	N	Q
10 PRINT "P=";				
15 INPUT P	100000			
20 PRINT "R="	100000			
25 INPUT R	100000	2		
30 PRINT	100000	2		
40 PRINT " N","POPULATION"	100000	2		
45 PRINT	100000	2		
50 LET N=0	100000	2	0	
60 LET Q=P*(1+R/100)↑N	100000	2	0	100000
70 PRINT N,Q	100000	2	0	100000
80 LET N=N+10	100000	2	10	100000
90 GO TO 60				
60 LET Q=P*(1+R/100)↑N	100000	2	10	121899.4
70 PRINT N,Q	100000	2	10	121899.4
80 LET N=N+10	100000	2	20	121899.4
90 GO TO 60				
60 LET Q=P*(1+R/100)↑N	100000	2	20	148594.7
70 PRINT N,Q	100000	2	20	148594.7
80 LET N=N+10	100000	2	30	148594.7
90 GO TO 60				

and so on, "forever".

P and R don't change

N changes every time
line 80 is obeyed

Q changes every time
line 60 is obeyed.

CROWDED

Our population table becomes more interesting if we include the number of people per square mile.

Easy! Simply modify the program on Page 26. Here are the changes.

```
27  PRINT "A=";                     A is the area in square miles.
28  INPUT A

40  PRINT " N", "POPULATION", "PEOPLE/SQ.MI."
70  PRINT N, Q, Q/A
```

Make the above changes and RUN the new program.

EARTH, 1970

P = 3.6E9 people

R = 2%

A = 57.2E6 square miles*

Your run should look like this.

```
RUN

P=?  3.6E9
R=?  2
A=?  57.2E6

N                POPULATION       PEOPLE/SQ.MI.

 0                3.600000E+09     62.93706
 10               4.388380E+09     76.71993
 20               5.349410E+09     93.52116
 30               6.520901E+09     114.0018
 40               7.948942E+09     138.9675
 50               9.689715E+09     169.4006
 60               1.181171E+10     206.4984
 70               1.439841E+10     251.7204
 80               1.755158E+10     306.8457
 90               2.139527E+10     374.0432
 100              2.608072E+10     455.9566    Getting crowded,
 .                    .                .
 .                    .                .
 .                    .                .

 200              1.836971E+11     3211.488    and still more crowded.

 and so on.
```

---

*Land area. New York Times Encyclopedic Almanac 1971, p. 324.

Make one more small change in the program. Change Line 70 to the following.

```
70  PRINT N, Q, INT(Q/A + .5)
```

Before we RUN it, let's LIST the program with all the changes mentioned on this and the preceding page.

```
LIST
10  PRINT "P=";
15  INPUT P
20  PRINT "R=";
25  INPUT R
27  PRINT "A=";
28  INPUT A
30  PRINT
40  PRINT " N","POPULATION","PEOPLE/SQ.MI."
45  PRINT
50  LET N=0
60  LET Q=P*(1+R/100)↑N
70  PRINT N,Q, INT(Q/A+.5)
80  LET N=N+10
90  GO TO 60
99  END
RUN

P=? 3.6E9
R=? 2
A=? 57.2E6

 N                 POPULATION      PEOPLE/SQ.MI.

 0                  3.600000E+09    63
 10                 4.388380E+09    77
 20                 5.349410E+09    94
 30                 6.520901E+09    114
 40                 7.948942E+09    139
 50                 9.689715E+09    169
 60                 1.181171E+10    206
 70                 1.439841E+10    252
 80                 1.755158E+10    307
```

Rounded to the nearest person.

and so on.

The INT function has the general form

INT(e)

where e is any BASIC expression. The INT function tells the computer to evaluate the expression and then compute the greatest integer that is less than or equal to the value of the expression.

EXAMPLES

INT(1) = 1
INT(-3) = -3
INT(3.999) = 3
INT(-3.14) = -4
INT(.001) = 0

INT(2) = 2
INT(3.6E9) = 3.600000E+9
INT(.987) = 0
INT(-999.05) = -1000
INT(-.001) = -1

More examples? Gather your own. RUN the following program.

```
10  PRINT "X=";
20  INPUT X
30  PRINT
40  PRINT "INT(X)=";INT(X),"INT(X+.5)=";INT(X+.5)
50  PRINT
60  GO TO 10
END
```

Be creative...choose both plain and fancy x's.

Your turn. Without using the computer, complete the following.

(1) INT(6.8) = ____________

(2) INT(6.8+.5) = ____________

(3) INT(6.2) = ____________

(4) INT(6.2+.5) = ____________

(5) INT(62.93706) = ________

(6) INT(62.93706+.5) = ________

(7) INT(206.4984) = ________

(8) INT(206.4984+.5) = ________

(9) INT(-3.9) = ____________

(10) INT(-3.6+.5) = ____________

(11) INT(-3.2) = ____________

(12) INT(-3.2+.5) = ____________

BEWARE OF MATHEMATICAL MODELS

We made an assumption and developed a *mathematical model* of population growth.

ASSUMPTION:	The increase in population each year is a percentage of the population at the beginning of the year. The percent increase, which we call n, does not change from year to year.
MATH MODEL:	If the initial population is P and the (constant) rate of increase is r% per year, then the population Q in n years is

$$Q = P(1 + r/100)^n$$

QUESTIONS

- Does our model really resemble real life?
- Can we use it to predict the future population of the U.S.? Of the Earth?
- How far into the future can we expect our model to provide reasonably accurate predictions? 10 years? 100 years? 1000 years?

The above questions lead to more questions.

- Is the rate of increase (r) really constant or is it increasing or decreasing?
- Is the rate the same for different regions of the Earth (e.g., North America, Asia, etc.)?
- Can we look more deeply into the mechanisms of population growth (birth rate, death rate, migration, life expectancy, fertility, etc.)?
- Where can we get more information?

The last question we can answer.

Population Reference Bureau, Inc., 1755 Massachusetts
Avenue, N.W., Washington, D.C., 20036.

CHAPTER 5

WORLDS OF IF

SON OF TABLE-MAKER

Earlier, we showed you a program to generate and print a table of

$$Q = P(1 + r/100)^n$$

for equally spaced values of n (n = 0, 10, 20, ...).

Here is the program again ... however, we have made one small change.

```
10  PRINT "P=";
15  INPUT P
20  PRINT "R=";
25  INPUT R
30  PRINT
40  PRINT " N","POPULATION"
45  PRINT
50  LET N=0
60  LET Q=P*(1+R/100)↑N
70  PRINT N,Q
80  LET N=N+10
90  IF N<=100 THEN 60 ←
99  END
RUN

P=? 205E6
R=? 1

 N                  POPULATION

 0                   2.050000E+08
 10                  2.264475E+08
 20                  2.501389E+08
 30                  2.763090E+08
 40                  3.052170E+08
 50                  3.371494E+08
 60                  3.724226E+08
 70                  4.113862E+08
 80                  4.544263E+08
 90                  5.019693E+08
 100                 5.544863E+08 ←

READY.
```

Here is the change. We use an IF statement instead of a GO TO statement. Read on!

After printing this line, the computer stops atuomatically, and types READY.

The statement

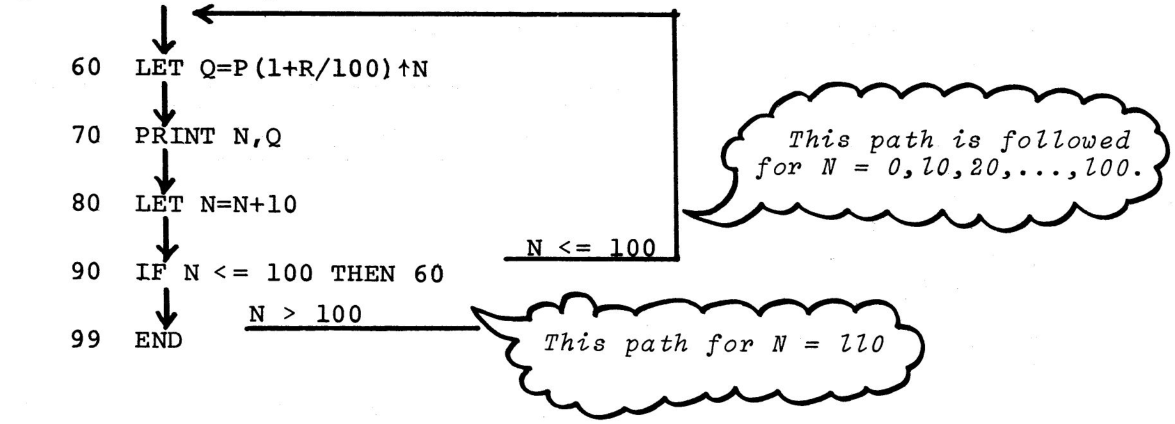
```
IF N <= 100 THEN 60
```

tells the computer:

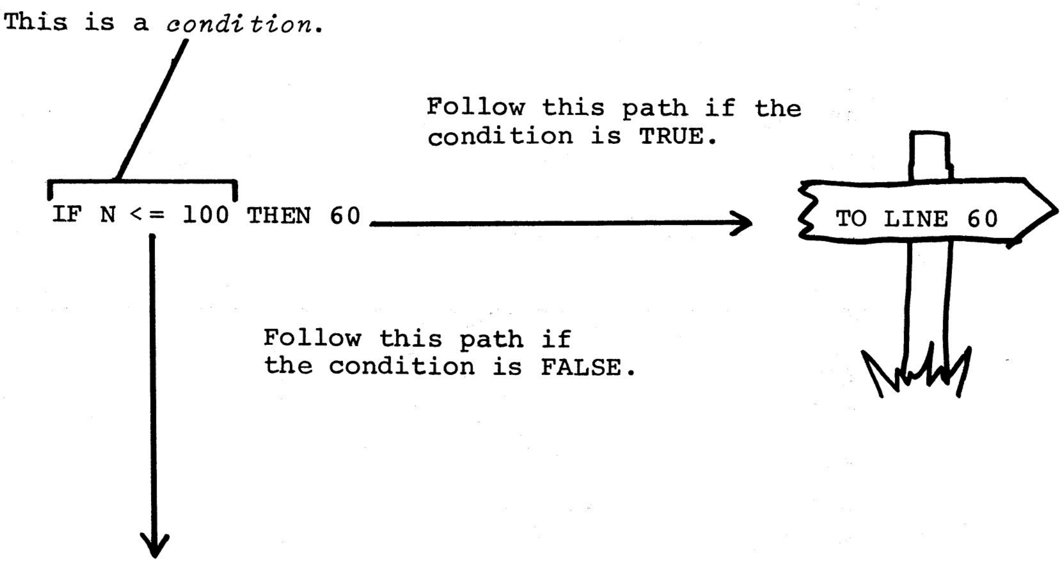
> IF THE VALUE OF N IS LESS THAN OR
> EQUAL TO 100 THEN GO TO LINE 60.

If the value of N is not less than or equal to 100 the computer goes to Line 99 (because 99 is the next higher line number after 90). Since Line 99 is an END statement, the computer stops.

Say it another way.



```
60  LET Q=P(1+R/100)↑N
70  PRINT N,Q
80  LET N=N+10
90  IF N <= 100 THEN 60
99  END
```



```
IF N <= 100 THEN 60
```

This is a *condition.*
Follow this path if the condition is TRUE.
TO LINE 60
Follow this path if the condition is FALSE.

Another view of the IF statement.

General Form:	IF	*condition*	THEN	*line number*
Example:	IF	K > 3	THEN	99

The condition K>3 is true for some values of K, false for other values.

Suppose K=4.	Then K>3 is TRUE and the computer will go to Line 99.
Suppose K=3	Then K>3 is FALSE and the computer will go to the line number that is next higher than the line number of the IF statement.

Let's use the above IF statement in a program.

```
10  LET K=1
20  IF K>3 THEN 99
30  PRINT K
40  LET K=K+1
50  GO TO 20
99  END
RUN

 1
 2
 3

READY
```

If K>3 is TRUE, go to Line 99.
If K>3 is FALSE, go to Line 30.

The computer stops automatically and types READY.

The condition in an IF statement is usually a math *relation* between two BASIC expressions. The permissible relations are shown in the table below.

RELATION	MATH SYMBOL	BASIC SYMBOL
equal to	=	=
less than	<	<
greater than	>	>
less than or equal to	≤	<=
greater than or equal to	≥	>=
not equal to	≠	<>

Let's *trace* the program. We will follow the computer as it RUNs the program and record the value of K after each statement is obeyed by the computer.

STATEMENT	K	REMARKS
10 LET K=1	1	
20 IF K>3 THEN 99	1	K>3 is FALSE. Go to Line 30.
30 PRINT K	1	Print the value of K.
40 LET K=K+1	2	Increase K by 1.
50 GO TO 20	2	Go around again.
20 IF K>3 THEN 99	2	K>3 is FALSE. Go to Line 30.
30 PRINT K	2	Print the value of K.
40 LET K=K+1	3	Increase K by 1.
50 GO TO 20	3	Go around again.
20 IF K>3 THEN 99	3	K>3 is FALSE. Go to Line 30.
30 PRINT K	3	Print the value of K.
40 LET K=K+1	4	Increase K by 1.
50 GO TO 20	4	Go around again.
20 IF K>3 THEN 99	4	K>3 is TRUE. Go to Line 99.
99 END		End of program. Computer prints READY and stops.

Your turn. Trace one of the following programs.

```
10  LET X=1
20  PRINT X
30  LET X=X+2
40  IF X< 8 THEN 20
99  END
```

```
10  LET C=4
20  PRINT C
30  IF C=0 THEN 99
40  LET C=C-1
50  GO TO 20
99  END
```

RACE TO OBLIVION

The following table shows the 1970 population and annual rate of population increase for various regions of the earth.*

REGION	POPULATION	RATE OF INCREASE(%)
AFRICA	344E6	2.6
ASIA	2056E6	2.3
NORTHERN AMERICA	228E6	1.1
LATIN AMERICA	283E6	2.9
EUROPE	462E6	0.8
U.S.S.R.	243E6	1.0
OCEANIA	19E6	2.0

The fastest growing region is Latin America and the slowest growing region is Europe. In 1970, the population of Europe was considerably more than the population of Latin America.

Assume: The growth rates for Europe and Latin America will remain constant.

Question: In what year will the population of Latin America become greater than the population of Europe?

Try this program.

```
10  LET N=1
20  LET E=462E6*(1+0.8/100)↑N
30  LET L=283E6*(1+2.9/100)↑N
40  IF L>E THEN 70
50  LET N=N+1
60  GO TO 20
70  PRINT "THE YEAR IS"; 1970+N
99  END
```

E *for Europe*
L *for Latin America*

---

*Data obtained from 1970 WORLD POPULATION DATA SHEET, Population Reference Bureau, Inc., 1755 Massachusetts Avenue, Washington, D. C., 20036.

A more general program. We enter the 1970 population and growth rate for each population. The computer computes and prints the year in which the *second* population overtakes the *first* population.

```
100  PRINT "FIRST POPULATION";
110  INPUT P1
120  PRINT "GROWTH RATE";
130  INPUT R1
140  PRINT
150  PRINT "SECOND POPULATION";
160  INPUT P2
170  PRINT "GROWTH RATE";
180  INPUT R2
190  PRINT
200  LET N=1
210  LET Q1=P1*(1+R1/100)↑N
220  LET Q2=P2*(1+R2/100)↑N
230  IF Q2>Q1 THEN 300
240  LET N=N+1
250  GO TO 210
300  PRINT "THE YEAR IS";1970+N
310  PRINT
320  GO TO 100
999  END
RUN
```

A BASIC variable may be a letter followed by a digit.

```
FIRST POPULATION?462E6
GROWTH RATE?0.8

SECOND POPULATION?283E6
GROWTH RATE?2.9

THE YEAR IS 1994

FIRST POPULATION?
```

Europe

Latin America

Your turn. But wait! Read on.

Assumptions. We assume that the first population is greater than the second population in 1970 but that the second population is growing more rapidly. Mathematically, we say that

P1 > P2 and R1 < R2

Try some data that violates one or both assumptions.

> Beware! You may have to stop the computer if you enter data for which the second population can't catch the first population.

We want to sneak in a new idea - checking for valid data. First, answer a few questions.

- → Will Northern America ever catch up with Latin America?
- → What happens if we enter Oceania as the first population and Asia as the second population?
- → Will Africa ever catch up with Latin America?

Remember, our assumptions are P1 > P2 and R1 < R2. Let's add some statements to the program to check the incoming data and reject data that violates one of the assumptions. The data violates the assumptions if

```
P1    <=    P2        or        R1    >=    R2
```

Add the following statements.

```
181  IF P1<=P2 THEN 184
182  IF R1>=R2 THEN 184
183  GO TO 190
184  PRINT "BAD DATA.  TRY AGAIN."
185  PRINT
186  GO TO 100
```

LIST the modified program.

RUN it. Use several sets of data. After all, there are 42 different ways to select a FIRST POPULATION and a SECOND POPULATION from the seven regions shown in the table.

If you have trouble figuring out how the program works, *trace it*.

YOUR TURN

Modify the program of Pages 38 and 39 so that results are printed as indicated below.

```
FIRST POPULATION? 462E6
GROWTH RATE?0.8

SECOND POPULATION? 283E6
GROWTH RATE?2.9


THE YEAR IS 1994
POPULATION ONE IS NOW 5.593641E+08
POPULATION TWO IS NOW 5.620247E+08

FIRST POPULATION?  and so on.
```

Modify the program so that the values of P1, R1, P2 and R2 are entered by means of READ and DATA statements. In fact, use the following DATA statements.

```
DATA 462E6, 0.8, 283E6, 2.9 ─┐
DATA 462E6, 0.8, 344E6, 2.6  ├── good data
DATA 462E6, 0.8, 243E6, 1.0  │
DATA 344E6, 2.6, 283E6, 2.9 ─┘
DATA 283E6, 2.9, 228E6, 1.1 ─┐
DATA 228E6, 1.1, 283E6, 2.9  ├── bad data
DATA 283E6, 2.9, 462E6, 0.8 ─┘
       P1    R1     P2    R2
```

For valid data, numerical results should be printed under the following heading. We show results corresponding to the data in the first (top) DATA statement above.

```
FIRST POP.     SECOND POP.     CATCHUP     NEW FIRST       NEW SECOND
  1970            1970          YEAR       POPULATION      POPULATION

4.62000E+08    2.83000E+06      1994       5.593641E+08    5.620247E+08

    P1             P2          1970+N         Q1              Q2
```

CHAPTER 6

CAROUSEL

1, 2, 3

Remember this program?

```
10  LET K=1
20  IF K>3 THEN 99
30  PRINT K
40  LET K=K+1
50  GO TO 20
99  END
RUN

 1
 2
 3

READY
```

You first saw it on Page 35.

It tells the computer to count to 3, then stop.

A new idea ... another way to count to 3.

```
10  FOR K=1 TO 3
20  PRINT K
30  NEXT K
99  END
RUN

 1
 2
 3

READY.
```

This is a FOR-NEXT loop.

Count to 3,

then stop.

You guessed it. This chapter is about FOR-NEXT loops.

A FOR-NEXT loop

- → begins with a FOR statement
- → ends with a NEXT statement.
- → usually includes one or more statements between the FOR and NEXT statements.

Eventually, we'll explain ... but first, more examples.

```
10  FOR K=0 TO 3
20  PRINT K
30  NEXT K
99  END
RUN

 1
 2
 3

READY.
```

```
10  FOR K=2 TO 7
20  PRINT K
30  NEXT K
99  END
RUN

 1
 2
 3
 4
 5
 6
 7

READY.
```

Got the idea? Try these. *Without using the computer*, complete each of the following by filling in the blanks.

```
10  FOR K=10 TO 13
20  PRINT K
30  NEXT K
99  END
RUN

________

________

________

________

READY.
```

```
10  FOR K=-1 TO 1
20  PRINT K
30  NEXT K
99  END
RUN

________

________

________

READY.
```

Have you noticed that all the programs on this page are identical except for Line 10? Good! Now check your work above by running the programs on the computer. Remember, you need change only Line 10 each time. Then experiment - check out the following FOR statements.

```
10  FOR K=5 TO 5              10  FOR K=3 TO 1

10  FOR K=1.5 TO 6.5          10  FOR K=0 TO 3.9

10  FOR K=1.25 TO 5           10  FOR K=1.25 TO 5.25
```

If any results puzzle you ... read on ... all will be revealed!

Let's use a FOR-NEXT loop in a program to read and print the population data shown in the table, top of Page 37 .

```
10  PRINT "POPULATION","GROWTH RATE"
20  PRINT

30  FOR K=1 TO 7
40  READ P,R
50  PRINT P,R
60  NEXT K
```

FOR-NEXT loop

```
90  DATA 344E6,2.6, 2056E6,2.3
91  DATA 228E6,1.1, 283E6,2.9
92  DATA 462E6,.8, 243E6,1, 19E6,2
```

Raw data from the table on Page 37 . There are seven *pairs* of numbers.

```
99  END

RUN

POPULATION    GROWTH RATE

 3.440000E+08  2.6
 2.056000E+09  2.3
 2.280000E+08  1.1
 2.830000E+08  2.9
 4.620000E+08  .8
 2.430000E+08  1
 1.900000E+07  2

READY
```

Each pair of numbers is read and printed, one pair per line.

Your turn. Modify the above program so that a RUN produces the following results.

```
RUN

POPULATION     GROWTH RATE    POPULATION
   1970            1970           2001

 3.440000E+08   2.6            7.623011E+08
 2.056000E+09   2.3            4.160711E+09
 2.280000E+08   1.1            3.200520E+08
 2.830000E+08   2.9            6.865356E+08
 4.620000E+08   .8             5.914503E+08
 2.430000E+08   1              3.308025E+08
 1.900000E+07   2              3.510418E+07

READY
```

Another modification ... here are the results.

```
RUN

POPULATION      GROWTH RATE     POPULATION
   1970            1970            2001

 3.440000E+08    2.6             7.623011E+08
 2.056000E+09    2.3             4.160711E+09
 2.280000E+08    1.1             3.200520E+08
 2.830000E+08    2.9             6.865356E+08
 4.620000E+08    .8              5.914503E+08
 2.430000E+08    1               3.308025E+08
 1.900000E+07    2               3.410418E+07

1970 WORLD POPULATION= 3.635000E+09
2001 WORLD POPULATION= 6.886957E+09

READY.
```

If you have trouble with the second modification, forget it for now ... read on ... return to the problem later.

- Check the 1970 WORLD POPULATION by adding the column of numbers under the 1970 POPULATION heading. Careful! The numbers are expressed in floating point notation.

```
Wrong way:     3.440000E+08
               2.056000E+09
               ------------
               5.496000E+17

Correct way:   3.440000E+08                       .344000E+09
              20.560000E+08          or          2.056000E+09
              -------------                      ------------
              24.000000E+08                      2.400000E+09
```

- Check 2001 WORLD POPULATION

- In 1970, the average growth for the WORLD POPULATION was 2%. Using the 1970 WORLD POPULATION and the 2% GROWTH RATE, compute the 2001 WORLD POPULATION (see pages 17 to 24). Is it the same as the result shown above? If not, why not?

(BEWARE OF MATHEMATICAL MODELS!)

FOR AND NEXT EXPLAINED

There are two general forms of the FOR statement. Here is the first general form with examples.

FOR	*variable*	=	*expression #1*	TO	*expression #2*
FOR	K	=	1	TO	3
FOR	X	=	0	TO	10
FOR	A	=	2	TO	5
FOR	J	=	1	TO	N
FOR	C	=	A	TO	B
FOR	Y	=	X	TO	X↑2
FOR	W	=	2*U+1	TO	3*U↑2-U

In other words

FOR ____________ = ________________ TO ____________________

any BASIC variable   any BASIC expression   any BASIC expression

Every FOR statement must have a NEXT statement that looks like this:

NEXT *variable*

same variable as the FOR statement

Got that? Here is another way to look at it.

```
FOR variable = expression #1 TO expression #2
 .
 .          ..........same variable
 .
NEXT variable
```

Here are some example-gathering programs for you to RUN. Try them ... you enter data as requested. Remember, to get out of a ? mode, type STOP and press RETURN.

The first four programs are identical except for Line 30.

```
10  PRINT "N=";
15  INPUT N
20  PRINT
30  FOR K=1 TO N
40  PRINT K;
50  NEXT K
60  PRINT
70  PRINT
80  GO TO 10
99  END
```

```
10  PRINT "N=";
15  INPUT N
20  PRINT
30  FOR K=1 TO 2*N
40  PRINT K;
50  NEXT K
60  PRINT
70  PRINT
80  GO TO 10
99  END
```

```
10  PRINT "N=";
15  INPUT N
20  PRINT
30  FOR K=1 TO N↑2
40  PRINT K;
50  NEXT K
60  PRINT
70  PRINT
80  GO TO 10
99  END
```

```
10  PRINT "N=";
15  INPUT N
20  PRINT
30  FOR K=N TO N↑2
40  PRINT K;
50  NEXT K
60  PRINT
70  PRINT
80  GO TO 10
99  END
```

```
10  PRINT "A=";
15  INPUT A
20  PRINT "B=";
25  INPUT B
30  PRINT
40  FOR X=A TO B
50  PRINT X
60  NEXT X
70  PRINT
80  GOTO 10
99  END
```

After running the program to the left, make the following changes and run it again.

```
35  LET S=0
45  LET S=S+X
50  PRINT X,S
```

Now can you do the modification on Page 44?

Ready for the second general form of the FOR statement?

First, two examples.

```
10  FOR K=1 TO 9 STEP 2
20  PRINT K;
30  NEXT K
99  END
RUN

 1    3    5    9
```

K is STEPped by 2.

```
10  FOR K=0 TO 10 STEP 3
20  PRINT K;
30  NEXT K
99  END
RUN

 0    3    6    9
```

K is STEPped by 3.

The second general form is

FOR	*variable*	=	*expression #1*	TO	*expression #2*	STEP	*expression #3*
FOR	K	=	1	TO	9	STEP	2
FOR	N	=	0	TO	100	STEP	10
FOR	X	=	A	TO	B	STEP	H
FOR	______	=	______	TO	______	STEP	______
	any variable		any expression		any expression		any expression

Now try this example gathering program.

```
10  PRINT "A,B,H=";
20  INPUT A,B, H
30  FOR X=A TO B STEP H
40  PRINT X;
50  NEXT X
60  PRINT
70  PRINT
80  GO TO 10
99  END
```

Be creative!

YOUR TURN

For the following programs, *predict* the results printed by the computer, then enter the program and RUN it to find out if you were correct.

```
10  LET S=0
20  FOR K=1 TO 7
30  READ P
40  LET S=S+P
50  NEXT K
60  PRINT S
90  DATA 344E6, 2056E6, 228E6
91  DATA 283E6, 462E6, 243E6, 19E6
99  END
```

This program is related to the second modification on Page 44 .

Rewrite the program on Page 33 . Use a FOR-NEXT loop to generate the values of N.

Rewrite the program on Page 33 so that the values of N go from A to B in steps of H. The values of A, B and H should be entered by means of INPUT statements.

CHAPTER 7

A PLACE FOR EVERYTHING

SUBSCRIPTED VARIABLES

Until now, we have used only *simple* BASIC variables. A simple variable consists of a letter (A - Z) or a letter followed by a single digit (0 - 9).

For example, the following are simple variables.

P R K P1 P2

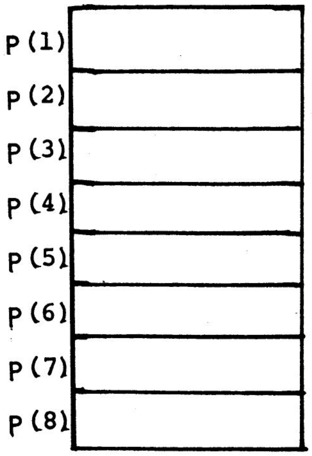
Now we want to introduce a new type of variable, called a *subscripted variable*.

Subscripted variable: P(5)
subscript: ↑

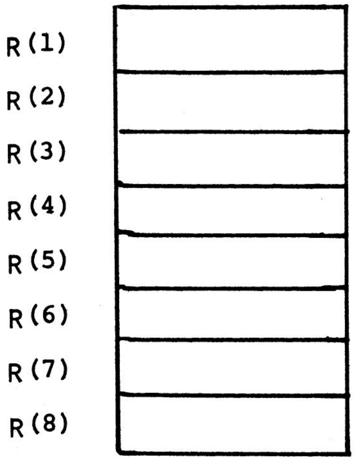
Say it like this: "P sub 5"

A subscripted variable names a location in the computer, a box, a place to store a number.

EIGHT SUBSCRIPTED VARIABLES



EIGHT SUBSCRIPTED VARIABLES



We can also use a variable as a subscript.

Subscripted variable: P(K)
subscript: ——↗

Call it "P sub K".

Below is the 1970 population and growth rate table, presented in a slightly different form from its previous appearance. Note that we have included World data as the last line in the table.

REGIONS	K	P(K) MILLIONS	R(K) %
AFRICA	1	344	2.6
ASIA	2	2056	2.3
N. AMERICA	3	228	1.1
L. AMERICA	4	283	2.9
EUROPE	5	462	0.8
U.S.S.R.	6	243	1.0
OCEANIA	7	19	2.0
WORLD	8	3635	2.0

P(K) is the population in *millions of people* for region K. For example, N. America is region 3.

P(3) = 228 MILLION PEOPLE

R(3) = 1.1%

Your turn. Complete the following.

P(2) = ______ MILLION PEOPLE

R(2) = ______%

P(8) = ______ MILLION PEOPLE

R(8) = ______%

Here is a program to print a table of K, P(K) and r(K). Our program includes REMARK statements which (we hope) make the program easier for *people* to read and understand. The computer ignores REMARK statements.

```
100  REMARK***READ VALUES INTO P(1),P(2),...,P(8)
110  FOR K=1 TO 8
120  READ P(K)
130  NEXT K
200  REMARK***READ VALUES INTO R(1),R(2),...,R(8)
210  FOR K=1 TO 8
220  READ R(K)
230  NEXT K
300  REMARK***PRINT THE TABLE
310  PRINT " K", " P(K)"," R(K)"
320  PRINT
330  FOR K=1 TO 8
340  PRINT K,P(K),R(K)
350  NEXT K
900  REMARK***DATA FOR P(1),P(2),...,P(8)
901  DATA 344,2056,228,283,462,243,29,3635
950  REMARK***DATA FOR R(1),R(2),...,R(8)
951  DATA 2.6,2.3,1.1,2.9,.8,1,2,2
999  END
RUN
 K              P(K)          R(K)

 1              344           2.6
 2              2056          2.3
 3              228           1.1
 4              283           2.9
 5              462           .8
 6              243           1
 7              19            2
 8              3635          2

READY.
```

remember, P(K) is *millions* of people.

In general, a REMARK statement

looks like this: REMARK *any remark*

or this: REM *any remark*

How does the program work?

(1) Lines 110-120 cause the computer to read the data in Line 901 into P(1) through P(8)

P(1)	344
P(2)	2056
P(3)	228
P(4)	283
P(5)	462
P(6)	243
P(7)	19
P(8)	3635

(2) Lines 210-230 cause the computer to read the data in Line 951 into R(1) through R(8).

R(1)	2.6
R(2)	2.3
R(3)	1.1
R(4)	2.9
R(5)	0.8
R(6)	1.0
R(7)	2.0
R(8)	2.0

(3) Lines 310-350 cause the computer to print a heading, then print 8 *rows* of numbers. Each row has three numbers: the values of K, the value of P(K) and the value of R(K).

Your turn. Let Q(K) be the population in region K, N years from 1970. That is, Q(K) is the population of region K in the year 2001. A RUN might look like the following.

```
RUN
 K            P(K)          R(K)          Q(K)

 1            344           2.6           762
 2            2056          2.3           4161
 3            228           1.1           320
 4            283           2.9           687
 5            462           .8            591
 6            243           1             331
 7             19           2             35
 8            3635          2             6716

READY.
```

GOSUB

A new idea ... subroutines ... and two new statements, GOSUB and RETURN. Let's use them in a rewrite of the program on Page 51 .

```
100  REMARK***MAIN PROGRAM
110  READ M
120  GOSUB 820
130  GOSUB 710
140  STOP

700  REMARK***SUBROUTINE TO PRINT TABLE OF K,P(K),R(K)
710  PRINT
720  PRINT " K"," P(K)"," R(K)"
730  PRINT
740  FOR K=1 TO M
750  PRINT K,P(K),R(K)
760  NEXT K
770  PRINT
780  RETURN

800  REMARK***SUBROUTINE TO READ DATA INTO P AND R
810  REMARK***READ VALUES INTO P(1),P(2),...,P(M)
820  FOR K=1 TO M
830  READ P(K)
840  NEXT K
850  REMARK***READ VALUES INTO R(1),R(2),...,R(M)
860  FOR K=1 TO M
870  READ R(K)
880  NEXT K
890  RETURN

900  REMARK***VALUE OF M
910  DATA 8                                       }
920  REMARK***VALUES OF P(1),P(2),...,P(M)        }
930  DATA 344,2056,228,283,462,243,19,3635        }  data base
940  REMARK***VALUES OF R(1),R(2),...,R(M)        }
950  DATA 2.6,2.3,1.1,2.9,.8,1,2,2                }

999  END
```

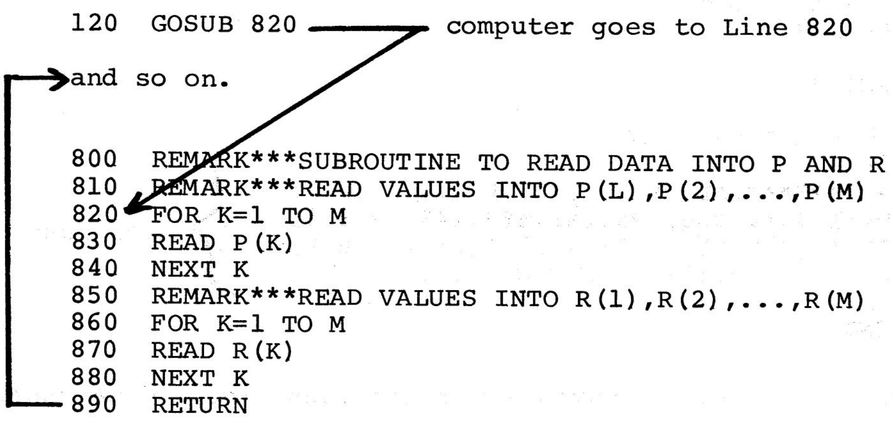
This program has a main program, two subroutines and a data base. The subroutine in Lines 800-890 is *called* by the GOSUB in Line 120. The subroutine in Lines 700-780 is *called* by the GOSUB in Line 130. More about that later.

Now, RUN the above program. The results should be the same as for the program on Page 51.

Let's look at the main program.

THE STATEMENT	TELLS THE COMPUTER ...
100 REMARK***MAIN PROGRAM	Ignore me. Move on to the next line.
110 READ M	Read the value of M. It is in the DATA statement, Line 910.
120 GOSUB 820	Do a subroutine, beginning at Line 820. On reaching a RETURN statement in the subroutine, return here, then move on to the next line.
130 GOSUB 710	Do a subroutine, beginning at Line 710. On reaching a RETURN statement in the subroutine, return here, then move on to the next line.
140 STOP	Stop the computer. (Computer types READY.)

Let's look at the GOSUB in Line 120 and the subroutine that it calls.



```
120   GOSUB 820 ——> computer goes to Line 820

and so on.

800   REMARK***SUBROUTINE TO READ DATA INTO P AND R
810   REMARK***READ VALUES INTO P(L),P(2),...,P(M)
820   FOR K=1 TO M
830   READ P(K)
840   NEXT K
850   REMARK***READ VALUES INTO R(1),R(2),...,R(M)
860   FOR K=1 TO M
870   READ R(K)
880   NEXT K
890   RETURN
```

RETURN sends the computer to the line following the GOSUB that sent the computer to the subroutine.

Next? We will use the two subroutines (Lines 700-890) and the data base (Lines 900-950) from the program on Page 53. We suggest that you punch this information into *paper tape* so that you can quickly re-enter it into the computer.

Paper tape? Have someone show you how to punch a program into paper tape and how to enter a program on paper tape into the computer. It's probably in your system user's guide, too.

Here is a new main program.

```
100  REMARK***FUTURE POPULATION, REGION K, YEAR Y
110  READ M
120  GOSUB 820
130  GOSUB 710
140  PRINT "REGION";
150  INPUT K
160  PRINT "YEAR";
170  INPUT Y
180  LET N=Y-1970
190  LET Q=P(K)*(1+R(K)/100)↑N
200  PRINT "POPULATION WILL BE";INT(Q+.5);"MILLION"
210  PRINT
220  PRINT
230  GO TO 140
```

Enter the above main program, then the two subroutines and the data base into the computer, then RUN the program. You should get results like the following.

```
 K               P(K)            R(K)

 1               344             2.6
 2               2056            2.3
 3               228             1.1
 4               283             2.9
 5               462             .8
 6               243             1
 7               19              2
 8               3635            2

REGION?1
YEAR?2001
POPULATION WILL BE 762 MILLION

REGION?__________
```

Which region do you want?

We think that you can figure out what the program is used for.

New main program (same old subroutines and data base).

```
100  REMARK***COMPARE TWO REGIONS FOR YEARS 1980,1990,2000
110  READ M
120  GOSUB 820
130  GOSUB 710
140  PRINT "LET'S COMPARE THE GROWTH OF TWO REGIONS, A AND B."
145  PRINT "WHEN I ASK, YOU ENTER THE REGION NUMBER (VALUE OF K)"
150  PRINT "FOR REGION A AND THE NUMBER FOR REGION B.  I WILL THEN"
155  PRINT "PRINT A TABLE SHOWING THE POPULATIONS OF REGIONS A"
160  PRINT "AND B FOR THE YEARS 1980, 1990 AND 2000."
170  PRINT
180  PRINT "REGION A";
190  INPUT A
200  PRINT "REGION B";
210  INPUT B
220  PRINT
230  PRINT "YEAR","REGION A","REGION B"
240  PRINT
250  LET N=10
260  LET Q1=P[A]*(1+R[A]/100)↑N
270  LET Q2=P[B]*(1+R[B]/100)↑N
280  PRINT 1970+N,INT(Q1+.5),INT(Q2+.5)
290  LET N=20
300  LET Q1=P[A]*(1+R[A]/100)↑N
310  LET Q2=P[B]*(1+R[B]/100)↑N
320  PRINT 1970+N,INT(Q1+.5),INT(Q2+.5)
330  LET N=30
340  LET Q1=P[A]*(1+R[A]/100)↑N
350  LET Q2=P[B]*(1+R[B]/100)↑N
360  PRINT 1970+N,INT(Q1+.5),INT(Q2+.5)
370  STOP
```

Put it all together and RUN it.

Then ...

- → Modify the program so that the table includes information for the years 2010 and 3000, (following the information for years 1980, 1990, and 2000. Examine the results for year 3000 carefully. Do you notice anything peculiar about the results? [Beware of mathematical models!]

- → In the main program above, did you notice that Lines 260-280 look like Lines 300-320, look like Lines 340-360? Does that give you any ideas about a subroutine and three GOSUBs?

CHAPTER 8

DEMOGRAPHY

DEMOGRAPHY ?

Yes, demography.

> *demography - the statistical science dealing with the distribution, density, vital statistics, etc. of population.*

Our table of population and growth rate for eight population regions is a set of demographic statistics. Here is another way to store it in the computer.

D is a table (matrix, array) of demographic data.
D is arranged in rows and columns, like this.

REGION	POPULATION		GROWTH RATE	
AFRICA	D(1,1)	344	D(1,2)	2.6
ASIA	D(2,1)	2056	D(2,2)	2.3
N. AMERICA	D(3,1)	228	D(3,2)	1.1
L. AMERICA	D(4,1)	283	D(4,2)	2.9
EUROPE	D(5,1)	462	D(5,2)	0.8
U.S.S.R.	D(6,1)	243	D(6,2)	1.0
OCEANIA	D(7,1)	19	D(7,2)	2.0
WORLD	D(8,1)	3635	D(8,2)	2.0

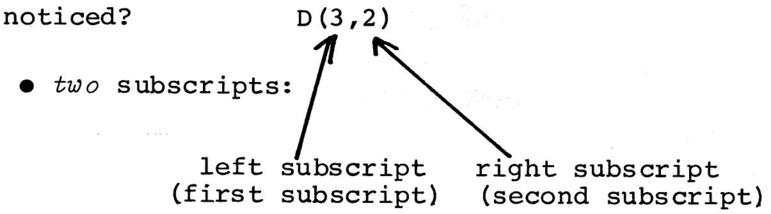
That's right. D(1,1) is a name of a box, location, place to store a number in the computer. So is (D(1,2) and D(2,1) and D(5,2) and D(6,1) and D(7,2) and ... . Complete the following.

→ What number is in D(4,1)?______________

→ What number is in D(8,2)?______________

→ The population of N. AMERICA is in D(____,____)

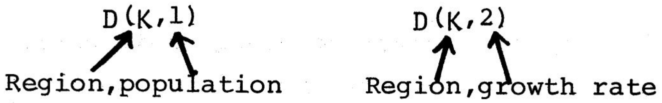
→ The growth rate of U.S.S.R. is in D(____,____) Have you noticed?



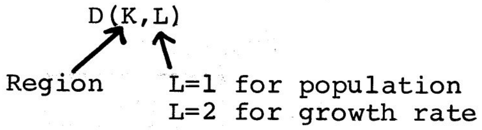
In our example on the preceding page

- The first subscript refers to the region. It may be 1,2,3,4,5,6,7,8. (But not 0 or 9)
- The second subscript refers to the *kind* of data, 1 for population and 2 for growth rate.

We can also use variable subscripts.



Or like this:



In Chapter 7, we worked with variables that had one subscript. In this chapter, we will work through many of the same problems but we will use variables that have two subscripts.

Let's start by rewriting the program on Page 53.

Compare the following program with the program on Page 53 .

```
100  REM ARK***MAIN PROGRAM
110  READ M
120  GOSUB 820
130  GOSUB 710
140  STOP
700  REM ARK***SUBROUTINE TO PRINT TABLE OF K,D(K,1),D(K,2)
710  PRINT
720  PRINT " REGION","POPULATION","GROWTH RATE"
730  PRINT
740  FOR K=1 TO M
750  PRINT K,D(K,1),D(K,2)
760  NEXT K
770  PRINT
780  RETURN
800  REM ARK***SUBROUTINE TO READ DATA INTO D TABLE
810  REM ARK***READ VALUES INTO D(1,1),...,D(M,1)
820  FOR K=1 TO M
830  READ D(K,1)
840  NEXT K
850  REM ARK***READ VALUES INTO D(1,2),...,D(M,2)
860  FOR K=1 TO M
870  READ D(K,2)
880  NEXT K
890  RETURN
900  REM ARK***VALUE OF M
910  DATA 8
920  REM ARK***VALUES OF D(1,1),...,D(M,1)
930  DATA 344,2056,228,283,462,243,19,3635
940  REM ARK***VALUES OF D(1,2)
950  DATA 2.6,2.3,1.1,2.9,.8,1,2,2
999  END
RUN

 REGION        POPULATION    GROWTH RATE

 1              344           2.6
 2              2056          2.3
 3              228           1.1
 4              283           2.9
 5              462           .8
 6              243           1
 7              19            2
 8              3635          2

READY.
```

There's always another way. Here is a different subroutine to read the data into the D table. In this subroutine

D(K,L)
Region L=1 for population
L=2 for growth rate.

```
800  REM ARK***SUBROUTINE TO READ DATA INTO D TABLE
810  FOR L=1 TO 2
820  FOR K=1 TO M
830  READ D(K,L)
840  NEXT K
850  NEXT L
860  RETURN
```

A loop within a loop.

The above subroutine must be entered at Line 810 instead of Line 820. So we change the main program as follows.

```
100  REM ARK***MAIN PROGRAM
110  READ M
120  GOSUB 810
130  GOSUB 710
140  STOP
```

Here is the change.

Now put it all together, main program, *new* subroutine to read data, old subroutine to print the table and old data base. RUN the complete program. The results should be the same as the results obtained using the program on Page 53.

CHANGING THE DATA BASE

We got our data base from the 1970 World Population Data Sheet. This data sheet lists data for 142 countries plus summaries for regional groups of countries. Suppose we set up a data base for all 142 countries. Might look like this:

COUNTRY	POPULATION	GROWTH RATE
1	P(1)	R(1)
2	P(2)	R(2)
3	P(3)	R(3)
.	.	.
.	.	.
.	.	.
142	P(142)	R(142)

where P(K) is the population for country K.
R(K) is the growth rate for country K.

Or, we could do it like this:

COUNTRY	POPULATION	GROWTH RATE
1	D(1,1)	D(1,2)
2	D(2,1)	D(2,2)
3	D(3,1)	D(3,2)
.	.	.
.	.	.
.	.	.
142	D(142,1)	D(142,2)

where D(K,1) is the population for country K.
D(K,2) is the growth rate for country K.

In general, suppose our data base includes statistics for M countries, (region, states, planets, etc.). If M is greater than 10, we must say so in our program. For example, we would modify our main program on Page 53 as follows.

```
100  REMARK***MAIN PROGRAM
110  READ M
115  DIM P(M), R(M)<-----------reserves space for
120  GOSUB 820                 the P(K)'s and R(K)'s.
130  GOSUB 810
140  STOP
```

The DIM (short for DIMension) tells the computer that

- Subscripts for P must be less than or equal to M.
- Subscripts for R must be less than or equal to M.

If we were using the program on Page 59 , we would add the following DIM statement to the main program.

```
115  DIM D(M,2)
```

This DIM statement tells the computer:

- Left subscript for D must be less than or equal to M.
- Right subscript for D must be less than or equal to 2.

YOUR TURN ... LAST TIME!

Here is your very own data base to play with.

COUNTRY	K	POPULATION	GROWTH RATE
AUSTRALIA	1	12.5	1.9
BRAZIL	2	93.0	2.8
CHINA	3	759.6	1.8
COSTA RICA	4	1.8	3.8
GERMANY, EAST	5	16.2	0.3
GERMANY, WEST	5	58.6	0.6
INDIA	6	554.6	2.6
JAPAN	7	103.5	1.1
MALTA	8	0.3	-0.8
MEXICO	10	50.7	3.4
NIGERIA	11	55.1	2.6
PHILLIPINES	12	38.1	3.4
U.S.S.R.	13	242.6	1.0
U.A.R.	14	33.9	0.5
UNITED KINGDOM	15	56.0	2.8
U.S.A.	16	205.2	1.0

Things to do.

- Set up a data base for use in the programs on Pages 53 and 59.
- Make any modifications you think are required in the programs, then load everything into the computer and RUN, RUN, RUN!
- Use the above data base in the programs on pages 55 and 56.
- Rewrite the programs on Pages 55 and 56 to use D(K,L) instead of P(K) and R(K). RUN them for the original data base and then again using the above data base.

ON BECOMING FLUENT

Now you can speak a little BASIC. Where do you go from here? Bear up! Help is on the way!

First, get help from your trustworthy computer supplier, DEC.

EduSystems User's Guides (available for EduSystem 10, 20, 30, and 50).

PDP-11 BASIC and RSTS-11 BASIC-PLUS User's Guides

BASIC Application Programs (Sets 1-4)

BASIC Simulation Programs

Volume I -	Biology, Earth Science
Volume II -	Chemistry
Volume III -	Mathematics
Volume IV -	Physics
Volume V -	Social Studies
Volume VI -	Teacher Assistance

Programming Languages

More help? We recommend these.

Basic BASIC, by James S. Coan. Hayden Book Company, Inc. New York, 1970.

BASIC Programming (2nd Edition) by John G. Kemeny and Thomas E. Kurtz. John Wiley & Sons, Inc., New York, 1970, 1971.

Population information?

Population Reference Bureau, Inc., 1755 Massachusetts Avenue, N.W., Washington, D. C. 20036.

The Population Bomb, by Dr. Paul R. Ehrlich, Ballantine Books, New York, 1968.

Population, by Roland Pressat. Penguin Books, Inc., Baltimore, 1970.

Population Control, edited by Anthony Allison. Penguin Books, Inc., Baltimore, 1970.